APLICACIONES CON ARDUINO: MANEJO DE DISPLAY DE 7 SEGMENTOS Y LED RGB
Aplicaciones con Arduino: Manejo de Display de 7 Segmentos y Led RGB

Ing. Esther Viridiana Vázquez Carmona
ev.vazquezc@gmail.com
Ing. Rodrigo Vázquez López
rodrigo_em2@hotmail.com
Dr. Juan Carlos Herrera Lozada
jcrls.ipn@gmail.com

Instituto Politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico en Cómputo (IPN-CIDETEC)

Resumen

En este documento se presenta una serie de diseños orientados a los sistemas embebidos, particularmente refiriéndose a la plataforma Arduino. Se exponen los principios de funcionamiento de un display de 7 segmentos para utilizarlo en un contador hexadecimal de 4 bits con decodificador incluido; adicionalmente se establece una comunicación serial para que Arduino reciba un valor numérico entre 0 y 15, proveniente de un teclado convencional para mostrarlo en un display a 7 segmentos en formato hexadecimal (0 – F). Finalmente se presenta la técnica de modulación por ancho de pulsos (PWM) integrándola con una comunicación serial para el control de un LED RGB lo que permitió diseñar un semáforo clásico. Los códigos implementados pueden reproducirse de manera sencilla para generar éstas u otras aplicaciones similares.

Introducción

Los Sistemas Embebidos tienen un sin fin de aplicaciones en la vida real, sin embargo, la mayoría de las aplicaciones requieren que los sistemas muestren información al usuario o se comuniquen con otros sistemas para enviar o recibir información. En este documento se abordarán cuatro aplicaciones básicas utilizando comunicación serial, display de 7 segmentos para el despliegue de información y el manejo de un LED RGB por medio de la técnica PWM, todas ellas utilizando la placa Arduino.

Display de 7 Segmentos

El display de 7 segmentos es un componente que sirve para representar números y letras, está compuesto por 7 LED tal y como se muestra en la figura 1, de esta forma controlando el encendido y apagado de cada segmento, es posible representar carácter que se requiera.

Existen dos tipos de display de 7 segmentos: ánodo común y cátodo común, su diferencia es la conexión que debemos implementar para encenderlos. En los 7 segmentos de Cátodo Común, el punto en común para todos los LED es el cátodo (GND), mientras que el Ánodo común el punto de referencia es Vcc (5 volt).

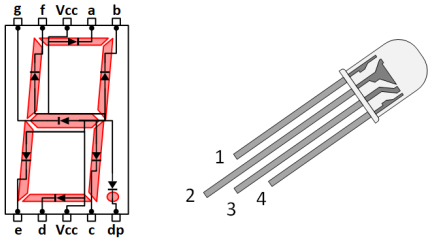

Figura 1. Display de 7 segmentos y Diodo LED RGB.

Comunicación Serial

Un puerto es la interface, física o virtual, que permite la comunicación entre dos dispositivos. Existe un medio de comunicación llamado puerto serie que envía la información mediante una secuencia de bits, a través de un Rx (recepción) y de Tx (transmisión).

Prácticamente todas las placas Arduino disponen al menos de una unidad UART (Universal Asynchronous Receiver-Transmitter). Algunos modelos de placas Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos de serie, lo que simplifica el proceso de conexión con una PC. Para la recepción y envío de datos, el IDE de Arduino cuenta con una utilidad llamada Monitor Serial, su uso es muy sencillo, y dispone de dos zonas, una que muestra los datos recibidos, y otra para enviarlos.

Diodo Emisor de Luz RGB

Los Diodos Emisor de Luz (LED) RGB, están compuestos por tres LEDs con totalidades rojo, azul y verde que al combinarse pueden producir más de 16 millones de tonos, sin embargo, no es posible obtener todos los colores, como el rosa, marrón, entre otros. Los LED RGB se pueden considerar como tres LEDs que se han unido entre sí.

Todos los LEDs constan de un ánodo y un cátodo. El ánodo es el polo positivo (generalmente asociado a 5V) y el cátodo es el polo negativo (generalmente asociado a GND). El voltaje de funcionamiento para cada color es aproximadamente 2.1V para el rojo y 3.3V para los colores verde y azul. Para variar las tonalidades del LED RGB, es necesario ajustar el voltaje en el ánodo o cátodo común según sea el tipo de LED con el que se está trabajando. La figura 1 muestra el LED RGB.

Modulación por ancho de pulsos (PWM)

La modulación por ancho de pulso (PWM) es una técnica de modulación que se basa en la variación de la anchura del pulso de una señal digital con base a una señal analógica dada. Cuando la señal analógica varía su amplitud, la anchura del pulso de la señal digital cambia.

Para obtener valores análogos variables, se modula ese ancho de pulso. Si repite este patrón de encendido-apagado lo suficientemente rápido con un LED, por ejemplo, el resultado es como si la señal fuera un voltaje constante entre 0 y 5v que controla el brillo del LED.

Desarrollo
Diseño de un contador ascendente

Una de las aplicaciones comunes para el uso del display de 7 segmentos, es mostrar valores numéricos provenientes de algún sistema embebido, estos valores pueden representar desde la lectura de temperatura de un sensor, hasta mostrar los números de turno en un banco. Una forma de observar su funcionamiento es conectar a la entrada un contador binario ascendente de 4 bits (módulo 16) con su respectivo decodificador, el objetivo es que dicha cuenta se muestre a través del display.

Para plantear dicho diseño dentro de la plataforma Arduino, el contador y el decodificador de 7 segmentos se implementan vía software. El display se conecta por medio de las salidas digitales al sistema embebido. El contador, genera una cuenta ascendente del 0-15, este se conecta al del decodificador de 7 segmentos en formato hexadecimal (0-F), para generar los valores de verdad requeridos para encender los segmentos correspondientes. El diagrama de conexiones y el resultado obtenido se muestra en la figura 2.

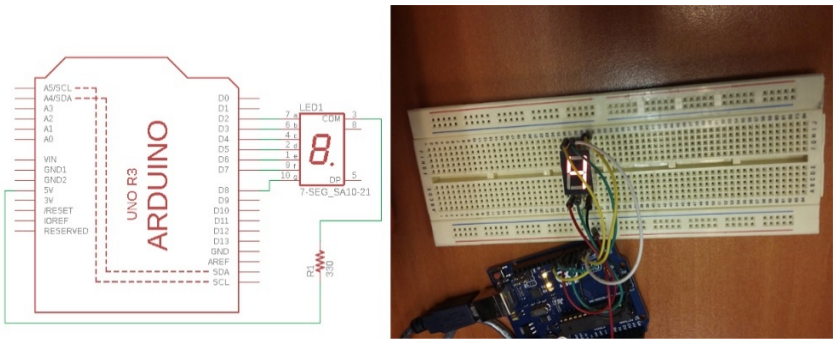

Figura 2. Diagrama de conexiones y resultado obtenido.

Despliegue de un valor utilizando comunicación serial

Existen aplicaciones donde los datos a mostrar por el sistema embebido provienen de otro dispositivo. Una manera simple de generar esta conexión es utilizando la comunicación serial. Para el siguiente diseño se implementa el uso de dicho recurso, el objetivo es mostrar en un display de 7 segmentos un dato recibido por la computadora.

Para este diseño, la UART del Arduino recibe un dato que proviene del teclado de una PC convencional, el dato es enviado al decodificador de 7 segmentos, para posteriormente ser mostrado en el display.

Cuando el puerto serial no recibe dato alguno, el programa se mantiene a la espera, una vez que recibe el primer byte, el carácter leído se almacena en una variable, este valor se transforma a un número entero y verifica que sea un valor entre 0 y 15. Si la condición se cumple, el dato es enviado al decodificador de 7 segmentos. De lo contrario, si la condición no se cumple, muestra un carácter 'H' en el display, indicando que el dato es erróneo.

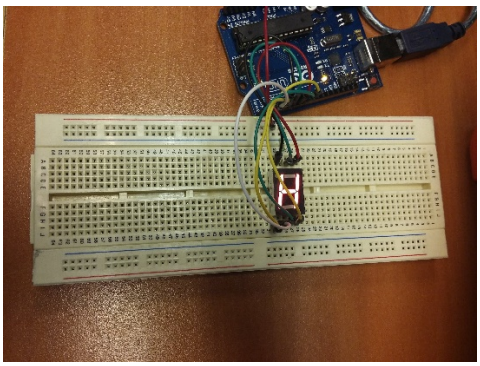

Figura 3. Despliegue de un dato erróneo.

Control de un LED RGB

En este diseño se plantea el control de tonalidades de un LED RGB utilizando comunicación serial, para enviar el valor RGB de los colores, que debe mostrar el LED.

En este diseño se plantea el control de tonalidades de un LED RGB utilizando comunicación serial, para enviar el valor RGB de los colores, que debe mostrar el LED.

El usuario escribe en el teclado tres números separados por comas que representan los valores RGB (en ese orden). Los valores introducidos son transformados a números enteros, cuando se recibe el carácter salto de línea, se manda a llamar a la función que se encargara de recibir esos valores para generar un color en el LED.

El diagrama de conexiones del circuito, mientras que la implementación del circuito se muestra en la figura.

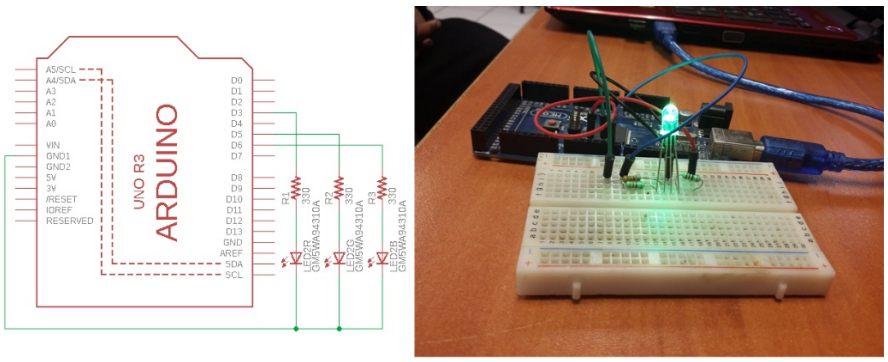

Figura 4. Diagrama de Conexiones y resultado obtenido.

Semáforo implementado con LED RGB

Un semáforo se encarga de regular el tránsito de vehículos. Está compuesto esencialmente por tres luces (verde, ámbar y rojo) programadas para que enciendan durante determinado tiempo.

Para su implementación podemos utilizar el mismo circuito del diseño anterior, únicamente cambia el programa, en el cual se fijan los tres colores que corresponden a un semáforo y se llama una función que genera el color respectivo en el LED. Además se incluye una función que genera el parpadeo de un color en específico. El resultado se muestra en la figura 5.

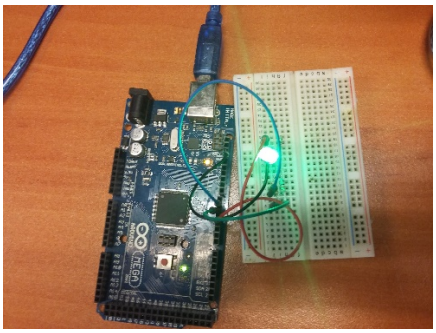

Figura 5. Semáforo con LED RGB.

Conclusiones

Arduino es una plataforma que se basa en una placa con un microcontrolador Atmel ATmega328 diseñada para el desarrollo de proyectos en electrónica que requieran prototipado rápido reduciendo el costo de componentes al disponer de toda una circuitería integrada y que da soporte a los procesos de adquisición y conectividad. En este trabajo se implementaron varios diseños básicos en la enseñanza de los sistemas embebidos. Se implementó un decodificador de 7 segmentos y un contador binario módulo 15 utilizando técnicas de programación, de esta forma disminuyó la cantidad de componentes.

Una de las herramientas más utilizadas en el mundo digital es la comunicación serial, ya que resulta sencilla su implementación en Arduino, sólo se requiere utilizar la conexión USB para enviar y recibir los datos. Arduino cuenta con puertos especiales para trabajar con la técnica de modulación PWM, por ejemplo, permite controlar la intensidad de un LED RGB lo que permitió realizar un semáforo convencional.

Los códigos que se implementaron se pueden reproducir sin cambios drásticos para generar estas mismas aplicaciones u otras similares.

Referencias

[1] Arduino. Arduino Reference. Accesed march 2018. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

[2] Display 7 segmentos. Accesed march 2018. [Online]. http://www.electrontools.com/Home/WP/2016/03/09/display-7-segmentos/

[3]

[4] Educachip. Accesed march 2018. [Online]. Available: http://www.educachip.com/led-rgb-arduino-anodo-comun/

[5] Funcionamiento de un LED RGB. Accesed march 2018. [Online]. Available: http://electronica-teoriaypractica.com/como-funciona-un-led-rgb/

[6] P. Marwedel, Embedded system design, 3rd~ed, Berlin, Germany: Springer, 2008.

[7] Philips. Lighting Philips. Accesed march 2018. [Online]. Available: http://www.lighting.philips.com.mx/soporte/soporte/preguntas-frecuentes/white-light-and-colour/what-does-rgb-led-mean#

[8] Puerto Serie. Accesed march 2018. [Online]. Available: https://www.luisllamas.es/arduino-puerto-serie/

[9] semáforos - funcionamiento. Accesed march 2018. [Online]. Available: http://www.eltiempo.com/archivo/documento/MAM-438910

Anexos

Código del contador

```
int pines[] = {2, 3, 4, 5, 6, 7, 8};
byte segmentos[16][7]={{1,1,1,1,1,1,0},
                       {0,1,1,0,0,0,0},
                       {1,1,0,1,1,0,1},
                       {1,1,1,1,0,0,1},
                       {0,1,1,0,0,1,1},
                       {1,0,1,1,0,1,1},
                       {1,0,1,1,1,1,1},
                       {1,1,1,0,0,0,0},
                       {1,1,1,1,1,1,1},
                       {1,1,1,1,0,1,1},
                       {1,1,1,0,1,1,1},
                       {0,0,1,1,1,1,1},
                       {0,0,0,1,1,0,1},
                       {0,1,1,1,1,0,1},
                       {1,0,0,1,1,1,1},
                       {1,0,0,0,1,1,1}}; // tabla de verdad del
decodificador 7 segmentos

//función para inicializar los pines conectados al display como salida
void setup() {
  for(int i=0;i<8;i++) {
    pinMode(pines[i],"OUTPUT");
  }
}

//function principal
void loop() {
  //Contador del 0 al 15 con una pausa de 1 seg
  for(int i=0;i<16;i++){
```

```
    dec_7seg(i);
    delay(1000);
  }
}

//Función que implementa el decodificador 7 segmentos
void dec_7seg(int n) {
  //Se extraen de la table de verdad los valores correspondientes a n y
los escribe en los puertos
  for(int i=0; i<8;i++){
      digitalWrite(pines[i],!segmentos[n][i]);// DISPLAY ÁNODO COMÚN
  }
}
```

Decodificador con comunicación serial

```
int pines[] = {2, 3, 4, 5, 6, 7, 8};
byte segmentos[17][7]={{1,1,1,1,1,1,0},
                       {0,1,1,0,0,0,0},
                       {1,1,0,1,1,0,1},
                       {1,1,1,1,0,0,1},
                       {0,1,1,0,0,1,1},
                       {1,0,1,1,0,1,1},
                       {1,0,1,1,1,1,1},
                       {1,1,1,0,0,0,0},
                       {1,1,1,1,1,1,1},
                       {1,1,1,1,0,1,1},
                       {1,1,1,0,1,1,1},
                       {0,0,1,1,1,1,1},
                       {0,0,0,1,1,0,1},
                       {0,1,1,1,1,0,1},
                       {1,0,0,1,1,1,1},
                       {1,0,0,0,1,1,1},//tabla de verdad del
decodificador 7 segmentos
                       {0,1,1,0,1,1,1}}; // carácter 'H' que representa
error o no válido

//función para incializar los puertos como salida y la comunicación
serial
void setup() {
  Serial.begin(9600);
  for(int i=0;i<8;i++) {
    pinMode(pines[i],"OUTPUT");
  }
}

//function principal
void loop() {
  //mientras el Arduino reciba datos
  while (Serial.available() > 0) {
    //transforma la cadena recibida a numero entero
    int numero = Serial.parseInt();
    Serial.println(numero,DEC);
   if(numero>=0 && numero <16) { //si el valor recibido está entre 0 y 15
lo manda al decodificador
      dec_7seg(numero);
```

```
      }
      else { //si el valor es diferente muestra 'H'
      {
        dec_7seg(16);
      }
    }
}

//Función que implementa el decodificador 7 segmentos
void dec_7seg(int n) {
  //Se extraen de la table de verdad los valores correspondientes a n y
los escribe en los puertos
  for(int i=0; i<8;i++){
      digitalWrite(pines[i],!segmentos[n][i]);// DISPLAY ANODO COMUN
  }
}
```

Control de un LED RGB

```
const int LEDred = 3;
const int LEDgreen = 5;
const int LEDblue = 6;

//inicializa los puertos conectados al LED RGB y la comunicacion serial
void setup() {
  Serial.begin(9600);
  pinMode(LEDred, OUTPUT);
  pinMode(LEDgreen, OUTPUT);
  pinMode(LEDblue, OUTPUT);
}

//function principal
void loop() {
  //Mientras Arduino reciba datos en el puerto serie
  while (Serial.available() > 0) {
    //convierte los numeros recibidos a enteros
    int R = Serial.parseInt();
    int G = Serial.parseInt();
    int B = Serial.parseInt();
    //cuando recibe un salto de linea se muestra el color
    if (Serial.read() == '\n') {
      mostrarColorLED(R,G,B);
    }
  }
}

//función para generar los valores PWM de un LED RGB
void mostrarColorLED(int R, int G, int B) {
  //Restringe los valores recibidos a enteros entre 0 y 255
  R = constrain(R, 0, 255);
  G = constrain(G, 0, 255);
  B = constrain(B, 0, 255);
  //Escribe los valores PWM recibidos
  analogWrite(LEDred, R);
  analogWrite(LEDgreen, G);
```

```
  analogWrite(LEDblue, B);
}
```

Semáforo RGB

```
const int redPin = 3;
const int greenPin = 5;
const int bluePin = 6;

//function para incializar los puertos conectados al LED RGB como salida
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

//function principal
void loop() {
  mostrarColorLED(0,255,0); //color verde
  delay(5000);
  parpadeoLedRGB(0,255,0,5); // parpadeo verde de 5 segundos
  mostrarColorLED(200,127,0); //color ambar
  delay(3000);
  mostrarColorLED(255,0,0); //color rojo
  delay(5000);
}

//Funcion para escribir los valores PWM de un LED RGB
void mostrarColorLED(int R, int G, int B) {
  //Restringe los valores recibidos a enteros entre 0 y 255
  R = constrain(R, 0, 255);
  G = constrain(G, 0, 255);
  B = constrain(B, 0, 255);
  //Escribe los valores PWM recibidos
  analogWrite(redPin, R);
  analogWrite(greenPin, G);
  analogWrite(bluePin, B);
}

//Función que implementa el parpadeo, recibe el color y la cantidad de
segundos de duracion
void parpadeoLedRGB(int R, int G, int B, int cantidad) {
  for(int i=0;i<cantidad;i++) {
    if(i%2==0) {
      //Si los segundos son pares apaga el LED
      mostrarColorLED(0,0,0);
    }
    else {
      //si son impares prende el LED con el color recibido
      mostrarColorLED(R,G,B);
    }
    //espera de un segundo
    delay(1000);
  }
}
```